🎨 Mini Project: Theme Toggle using React Context API

💡 Goal:

User ek button se theme (Light/Dark) toggle kare — bina prop drilling ke, using `useContext`.

🧳 Concept Flow:

```
App (Provider)
 └── ChildA
     └── ChildB
         └── ChildC (🎯 Button to change theme)
```

📌 Why useContext here?

- `theme` and `setTheme` ko direct deeply nested component (ChildC) tak le jaana hai
- Prop drilling avoid karke neat aur scalable structure milta hai

🔨 Steps to Build:

🧾 Step 1: Create Context

```
const ThemeContext = createContext();
```

🧾 Step 2: Provide Value (theme + setTheme)

```
<ThemeContext.Provider value={{ theme, setTheme }}>
  {/* children */}
```

```
</ThemeContext.Provider>
```

🧾 Step 3: Wrap UI that needs to reflect theme

```
<div id="content" style={{ backgroundColor: theme === 'Light' ? 'aquamarin
e' : 'black' }}>
  <ChildA />
</div>
```

🧾 Step 4: Consume & Toggle from Deep Child (ChildC)

```
const { theme, setTheme } = useContext(ThemeContext);

function toggleTheme() {
  setTheme(theme === "Light" ? "Dark" : "Light");
}
```

🧠 Full Code Walkthrough

📁 App.jsx

```
import { useState, createContext } from 'react';
import './App.css';
import ChildA from './components/ChildA';

const ThemeContext = createContext();

function App() {
  const [theme, setTheme] = useState('Light');

  return (
    <ThemeContext.Provider value={{ theme, setTheme }}>
```

```
      <div
        id="content"
        style={{
          backgroundColor: theme === 'Light' ? 'aquamarine' : 'black',
          minHeight: '100vh',
        }}
      >
        <ChildA />
      </div>
    </ThemeContext.Provider>
  );
}

export default App;
export { ThemeContext };
```

📁 ChildA.jsx

```
import ChildB from './ChildB';

const ChildA = () => {
  return <ChildB />;
};

export default ChildA;
```

📁 ChildB.jsx

```
import ChildC from './ChildC';

const ChildB = () => {
  return <ChildC />;
};
```

```
export default ChildB;
```

📁 ChildC.jsx

```
import { useContext } from 'react';
import { ThemeContext } from '../App';

const ChildC = () => {
  const { theme, setTheme } = useContext(ThemeContext);

  function toggleTheme() {
    setTheme(theme === "Light" ? "Dark" : "Light");
  }

  return (
    <div>
      <button onClick={toggleTheme}>CHANGE THEME</button>
    </div>
  );
};

export default ChildC;
```

📦 Output:

> 🟢 Background color changes between aquamarine (Light) and black (Dark) when button is clicked.

📋 Summary Table:

Concept	Code

Create Context	`const ThemeContext = createContext();`
Provider	`<ThemeContext.Provider value={{theme, setTheme}}>`
useContext Hook	`const {theme, setTheme} = useContext(ThemeContext);`
Theme Logic	`setTheme(theme === 'Light' ? 'Dark' : 'Light')`

✅ Real-World Use Cases:

- Light/Dark Theme Toggle 🌓
- Language change (i18n) 🌍
- User authentication status 🔐
- Sidebar open/close toggle 🧩
